REMARKS

In the Office Action, the Examiner rejected Claims 1-24, which are all of the pending claims, under 35 U.S.C. 103 as being unpatentable over U.S. Patent 6,219,690 (Slingwine). Claims 10, 11 and 13-17 were further rejected under 35 U.S.C. 101 as directed to non-statutory subject matter; and claims 10-17 were also rejected under 35 U.S.C. 112, second paragraph. In addition, Claims 6, 14 and 21 were rejected under the doctrine of obviousness-type double patenting as being unpatentable over claims of copending application no. 11/227,761.

Independent Claims 1, 10 and 18 are being amended to better define the subject matters of thee claims. Claims 10 and 17 are being amended to address the rejections of Claims 10-17 under 35 U.S.C. 112. Claims 5, 13 and 23 are being amended to remove features being added to Claims 1, 10 and 18 respectively, and Claim 7 is being amended to keep the language of these claims consistent with the language of amended Claim 1.

The rejection of Claims 10, 11 and 13-17 under 35 U.S.C. 101 is respectfully traversed. Also, Applicants request that the double patenting rejection of Claims 6, 14 and 21 be held in abeyance pending an indication that the claims patentably distinguish over the prior art.

Applicants will first address the rejections under 35 U.S.C. 101 and 112, and then discuss the rejection of the claims under 35 U.S.C. 103.

With respect to the rejection of Claims 10, 11 and 13-17 under 35 U.S.C. 101, the Examiner, in the Office Action, argued that these claims are directed to non-statutory subject matter because the claims fail to provide hardware support to carry out the software functionality described in the claims.

Claims 11 and 13-17 are all dependent from Claim 10. This Claim 10 expressly sets forth "a computer system," and the software functionality described in Claims 10, 11 and 13-17 can all be done on this computer system. More specifically, the preamble of Claim 10 sets forth "A system for swapping source code in a computer system," and the body of Claim 10 sets means that are used to achieve that swapping. The computer system described in the preamble of Claim 10 is hardware that can be used to perform the functions needed to swap the code in the manner set forth in Claim 10, and also to perform all of the functions set forth in Claims 11 and 13-17.

In view of the foregoing, all of Claims 10, 11 and 13-17 include hardware to carry out the claimed functions, and all of these claims are directed to statutory subject matter within the meaning of 35 U.S.C. 101. The Examiner is thus asked to reconsider and to withdraw the rejection of Claims 10, 11 and 13-17 under 35 U.S.C. 101.

Claims 10-17 were rejected under 35 U.S.C. 112, second paragraph, as being incomplete. In this rejection, the Examiner noted that the preamble of Claim 10 refers to "swapping source code." The Examiner argued that source code is nowhere disclosed in the Disclosure, and that there appears to be a gap of teaching or undefined relationship between the source code and the replacing step in Claim 10.

Applicants respectfully note that the specification, in paragraph 13, lines 4-6 expressly states that the invention may be used to replace first source code with a new source code component. Further, this may be done in the manner taught in the detailed description section of the application. In addition, Claim 10 is being amended to change reference to "code" to "source code," and Claim 10 is also being amended, as discussed in more detail below, to recite additional features of the swapping mechanism. With these changes, it is believed that Claim 10

describes a complete and operative system for swapping source code, and the Examiner is also asked to reconsider and to withdraw the rejection of Claims 10-17 under 35 U.S.C. 112, second paragraph, as being incomplete.

Claim 17 was further rejected under 35 U.S.C. 112, second paragraph, as being indefinite. In making this rejection, the Examiner noted, in the Office Action, that the claim recites "A method according to Claim 10," while Claim 10 is a system claim.

Line 1 of Claim 17 is being amended to change "method" to "system." In this way, Claim 17 is now consistent with Claim 10, and the Examiner is asked to reconsider and to withdraw the rejection of Claim 17 under 35 U.S.C. 112 as being indefinite.

In addition to the foregoing, all of Claims 1-24 patentably distinguish over the prior art. The Examiner is thus also requested to reconsider and to withdraw the rejection of Claims 1-24 under 35 U.S.C. 103, and to allow these claims.

Generally, Claims 1-24 patentably distinguish over the prior art because the prior art does not disclose or render obvious the way in which code or a code component of a computer system is swapped, as described in the independent Claims 1, 10 and 18. More specifically, the prior art does not disclose or render obvious establishing a quiescent state for a first code component, transferring state from the first code component to a new code component, and swapping the new code component for the first code component, as described in those independent claims.

As discussed in the present application, operating systems are large and complex, and must satisfy a number of difficult requirements. For instance, these systems are expected to run across a varied set of hardware platforms, are expected to stay up for long periods of time, and are expected to stay up to date with the latest fixes. Further, these operating systems serve an increasingly divergent set of application workloads.

In the past, operating systems programmers have used several approaches to attempt to achieve some of these goals, and there is a large body of prior work focusing on the downloading and dynamic binding of new components. There has, though been less work on swapping of transparent scalable components in an active system. In particular, the prior art has not been able to provide a generic hot-swapping capability for operating systems that allows them to activate new code without stopping the service, while also performing effectively under a varying set of workloads and while ensuring continuous availability.

The present invention effectively addresses this need.

Generally, this is done by allowing new code, which has been downloaded to fix or upgrade a particular service, to be enabled and activated in a computer operating system without having to bring down the system or the service.

More specifically, in one embodiment, the invention provides a method of replacing a first code component in a computer system with a new code component while the operating system of the computer remains active and while that operating system provides continual availability to hardware resources by applications operational in the computer system.

In this method, an instance of a new code component to replace the first code component is instantiated. A quiescent state is established for the first code component, state is transferred from the first code component to the new code component, and the new code component is then swapped for the first code component. This swapping includes the steps of identifying references to the first code component, and replacing the identified references to that first code component with references to the new code component.

The prior art does not disclose or render obvious establishing a quiescent state for a first code component, transferring state from the first code component to a new code component, and swapping the new code component for the first code component, as described above.

In particular, Slingwine, et al, which is the only reference relied on by the Examiner to reject the claims. Describes a procedure for updating data. This procedure allows concurrent reading and updating data while maintaining data coherency.

In the Slingwine, et al. procedure, a mutual-exclusion mechanism tracks a thread execution history to determine safe times for processing a current generation of data updates while a next generation of data updates is concurrently being saved. A summary of thread activity tracks which threads have passed through a quiescent state after the current generation of updates was started. When the last thread related to the current generation passes through a quiescent state, the summary of thread activity signals a callback processor that it is safe to end the current generation of updates. This callback processor then processes and erases all updates in the current generation, and the next generation of updates then becomes the current generation of updates.

There is a very important general difference between the present invention and the procedure disclosed in Slingwine, et al – the present invention is directed to swapping code, while Slingwine, et al. is directed to updating data.

This general difference is reflected in a number of more specific differences between the present invention and the method and system disclosed in Slingwine, et al.

One very important difference is that Slingwine, et al. does not show transferring state from the old code to the new code. Moreover, there is no reason to do this in Slingwine, et al. because in the procedure disclosed therein, it does not matter what the old data is – that old data

is being replaced with the new data. With the present invention, in contrast, when code is replacing code, it is critical that the new code have the state of the old code.

Independent Claims 1, 10 and 18 are being amended to emphasize the above-discussed feature of the present invention. In particular, each of these claims is being amended to describe the feature of establishing a quiescent state for a first code component, transferring state from the first code component to a new code component, and swapping the new code component for the first code component.

These features are of utility because they can be used, as taught in the present application, to help provide a maintainable well-performing code for a divergent set of applications.

The other references of record have been reviewed, and these other references, whether considered individually or in combination, also do not show or render obvious this aspect of the present invention.

Because of the above-discussed differences between Claims 1, 10 and 18 and the prior art, and because of the advantages associated with these differences, Claims 1, 10 and 18 patentably distinguish over the prior art and are allowable. Claims 2-9 are dependent from Claim 1 and are allowable therewith. Likewise, Claims 11-17 are dependent from Claim 10 and are allowable therewith; and Claims 19-24 are dependent from, and are allowable with, Claim 18. The Examiner is, accordingly, respectfully asked to reconsider and to withdraw the rejection of Claims 1-24 under 35 U.S.C. 103 and to allow these claims.

For the reasons advanced above, the Examiner is asked to reconsider and to withdraw the rejections of Claims 10, 11 and 13-17 under 35 U.S.C. 101 and the rejections of Claims 10-17 under 35 U.S.C. 112. The Examiner is also asked to reconsider and to withdraw the rejection of Claims 1-24 under 35 U.S.C. 103, and to allow these claims. If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully submitted,

John S. Sensny

John S. Sensny
Registration No. 28,757
Attorney for Applicants

SCULLY, SCOTT, MURPHY & PRESSER, P.C.
400 Garden City Plaza – Suite 300
Garden City, New York 11530
(516) 742-4343

JSS:jy